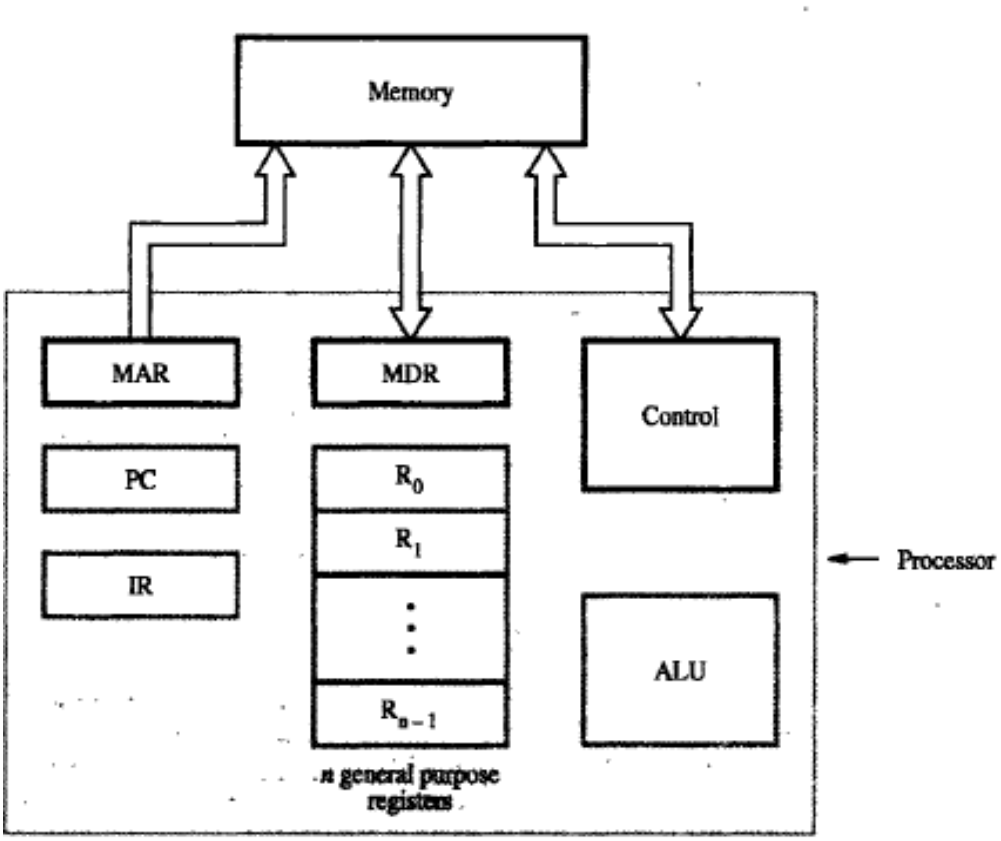
Module 3

Basic Structure of computers

➢ *BASIC OPERATIONAL CONCEPTS*

The Processor consists of different types of registers.

1. MAR (Memory Address Register)
2. MDR (Memory Data Register)
3. Control Unit
4. PC (Program Counter)
5. Register Array or Register File
6. IR (Instruction Register)
7. ALU (Arithmetic and Logic Unit



The connection between Memory and Processor is as shown in the figure

The functions of these registers are as follows

1. MAR

It establishes communication between Memory and Processor. It stores the address of the Memory Location

2. MDR

It also establishes communication between Memory and the Processor. It stores the contents of the memory location (data or operand) to be Read or write into the memory

3. CONTROL UNIT

It controls the data transfer operations between memory and the processor. It controls the data transfer operations between I/O and processor. It generates control signals for Memory and I/O devices.

4. PC (PROGRAM COUNTER)

It is a special purpose register used to hold the address of the next instruction to be executed. The contents of PC are incremented by 1 or 2 or 4 after either instruction or data fetched from memory The contents of pc are incremented by 1 for 8 bit CPU, 2 for 16 bit CPU and for 4 for 32 bit CPU

5. REGISTER ARRAY

The structure of register file is as shown in the above figure. It consists of set of registers. A register is defined as group of flip flops. Each flip flop is designed to store 1 bit of data. It is a storage element.

6. IR (INSTRUCTION REGISTER

It holds the instruction to be executed. It output is available to the control units.

7. ALU (ARITHMETIC and LOGIC UNIT)

It performs arithmetic and logical operations on given data.

STEPS FOR INSTRUCTION EXECUTION

The steps for instruction execution are as follows

1. Fetch the instruction from memory into the IR.
2. Decode the instruction
3. Access the Memory Operand
4. Access the Register Operand
5. Perform the operation according to the Operation Code.
6. Store the result into the Destination Memory location or Destination Register.

➢ *BUS STRUCTURE*

Bus: It is defined as set of parallel wires used for data communication. Each wire carries 1 bit of data. There are 3 of buses, namely

1. Address bus
2. Data bus and
3. Control bus.

1. ***Address bus :***

It is unidirectional.

The CPU sends the address of an I/O device or Memory device by means of this bus.

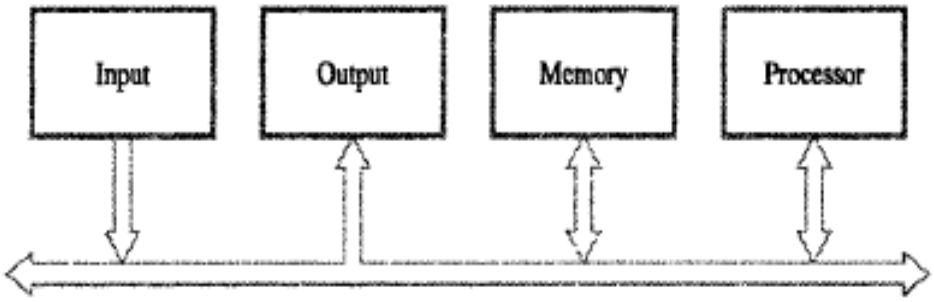
2. ***Data bus***

It is a bidirectional bus.

The CPU sends data from Memory to CPU and vice versa as well as from I/O to CPU and vice versa by means of this bus.

3. ***Control bus:***

This bus carries control signals for Memory and I/O devices.



Single bus organization

- The I/O devices, Memory and CPU are connected to this bus is as shown in the figure.
- It establishes communication between two devices.

Features of Single bus organization are

- Less Expensive
- Flexible to connect I/O devices.
- Poor performance due to single bus.

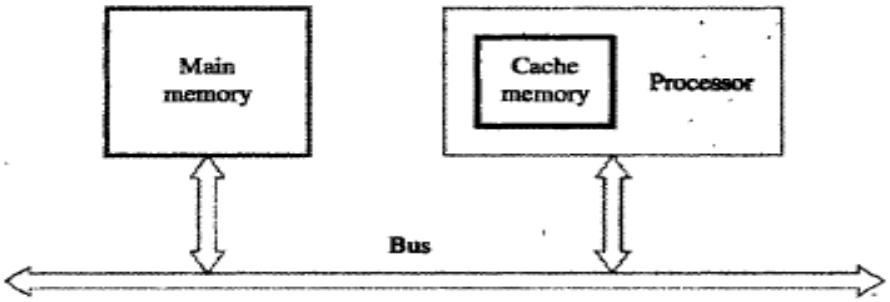
There is a variation in the devices connected to this bus in terms of speed of operation. To provide the synchronization between two devices, a buffer register is attached to each device. It holds the data temporarily during the data transfer between two devices.

➢ *PERFORMANCE*

- The performance of a Computer System is based on hardware design of the processor and the instruction set of the processors. To obtain high performance of computer system it is necessary to reduce the execution time of the processor.
- Execution time: It is defined as total time required executing one complete program.
- The performance of the processor is inversely proportional to execution time of the processor.
- More performance = Less Execution time.
- Less Performance = More Execution time.

The Performance of the Computer System is based on the following factors

.CACHE MEMORY: It is defined as a *fast access memory* located in between CPU and Memory. It is part of the processor as shown in the fig



The processor needs more time to read the data and instructions from main memory because main memory is away from the processor as shown in the figure. Hence it slowdown the performance of the system.

The processor needs less time to read the data and instructions from Cache Memory because it is part of the processor. Hence it improves the performance of the system.

***PROCESSOR CLOCK*:**

The processor circuits are controlled by timing signals called as Clock.It defines constant time intervals and are called as Clock Cycles. To execute one instruction there are 3 basic steps namely

1. Fetch
2. Decode
3. Execute.

The processor uses one clock cycle to perform one operation as shown in the figure

Clock Cycle → T1 T2 T3

Instruction → Fetch Decode Execute

The performance of the processor depends on the length of the clock cycle. To obtain high performance reduce the length of the clock cycle. Let ' P ' be the number of clock cycles generated by the Processor and ' R ' be the Clock rate .

The Clock rate is inversely proportional to the number of clock cycles.

i.e R = 1/P.

Ex 1: R= 500MHZ, P=?

$1/500 = 0.002 * 10^{-6} = 2ns$

Ex 2 : R=1250 MHZ , P = ?

$1/1250 = 0.0008 * 10^{-6} = 0.8$ ns

BASIC PERFORMANCE EQUATION

Let ' T ' be *total time* required to execute the program.

Let 'N ' be the *number of instructions* contained in the program. To execute one instruction there are 3 steps namely 1. Fetch 2. Decode 3. Execute

Let ' S ' be the *average number of steps* required to one instruction.

Let ' R' be number of clock cycles per second generated by the processor to execute one program. Hence Processor Execution Time is given by

T = N * S / R

This equation is called as Basic Performance Equation.

For the programmer the value of T is important. To obtain high performance it is necessary to reduce the values of N and S and to increase the value of R

CLOCK RATE

- Improving the integrating –circuit (IC) technology makes logic circuits faster, which reduces the time needed to complete a basic step. this allows the clock period 'p' to be reduced and the clock rate 'R' to be increased.
- Reducing the amount of processing done in one basic step also makes it possible to reduce the clock period.

PERFORMANCE MEASUREMENT

- The computer community adopted the idea of measuring computer performance using benchmark programs.
- The performance measure is the time it takes a computer to execute given benchmark
- An organization called system performance evaluation corporation(SPEC) selects and publishes programs for different application domains.
- It also provides many test results for commercially available computers.
- This was developed in the year 1995 and modified in the year 2000, respectively called as SPEC95 and SPEC2000.
- Programs are selected from various fields like games, database, numerical calculations.
- The program is compiled for the computer under test, and running time on that computer is measured.
- The same program is compiled and run on one computer selected as a reference.
- For SPEC95, the reference is the SUN SPARC station, for SPEC2000, the reference computer is an ULTRA SPARC workstation.

- *Spec rating :*
  Spec rating is computed as follows

$$\text{spec rating} = \frac{\text{running time on the reference computer}}{\text{running time on the computer under test}}$$

- Spec rating of 50 means that the computer under test is 50 times as fast as the reference computer for that particular benchmark.

- The test is repeated for all programs in the spec suit, and the geometric mean of the results is computed.
- let $spec_i$ be the rating for program i in the suite.
- The overall spec rating for the computer is given by
-

$$\text{Spec Rating} = \left(\prod_{i=1}^{n} SPEC_i\right)^{1/n}$$

Where , n is the number of programs in the suite.

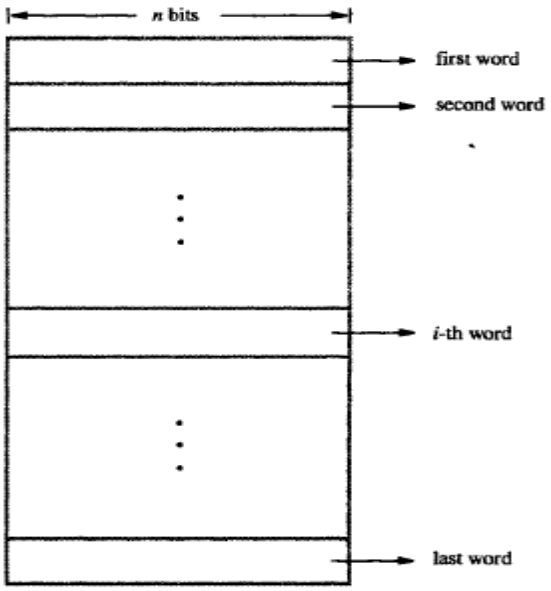
The computer providing higher performance have higher SPEC rating.

➢ *MEMORY LOCATIONS AND ADDRESSES*

1. Memory is a storage device. It is used to store character operands, data operands and instructions.
2. It consists of number of semiconductor cells and each cell holds 1 bit of information.A group of 8 bits is called as byte and a group of 16-64 bits is called as word.

World length = 16 for 16 bit CPU and World length = 8 for 8 bit CPU. It is defined as number of bits in the byte or word.

- Memory is organized in terms of bytes or words.
- The organization of memory for 32 bit processor is as shown in the fig.



Memory words

The contents of memory location can be accessed for read and write operation either by specifying address of the memory location or by name of the memory location.

Address space : It is defined as number of bytes accessible to CPU and it depends on the number of address lines.

Let 'x' be the number of address lines.

Hence number of bytes accessible to CPU = $2^{X,}$ where x is the number of address lines.

Ex:1 Consider 8085 CPU. It consists of 8 data lines and 16 address lines .Hence number of bytes accessible to 8085 CPU = 2^{16}=65,536 bytes.

BYTE ADDRESSABILITY

The computer performs ALU operations on 3 quantities namely bit, byte and word. It is impractical to assign addresses for 1 bit of information. Hence for practical reasons it is necessary to assign the addresses for successive bytes.

- Hence Byte Addressability is the process of assignment of address to successive bytes of the memory. The successive bytes have the addresses 1, 2, 3, 4…….2^n-1.

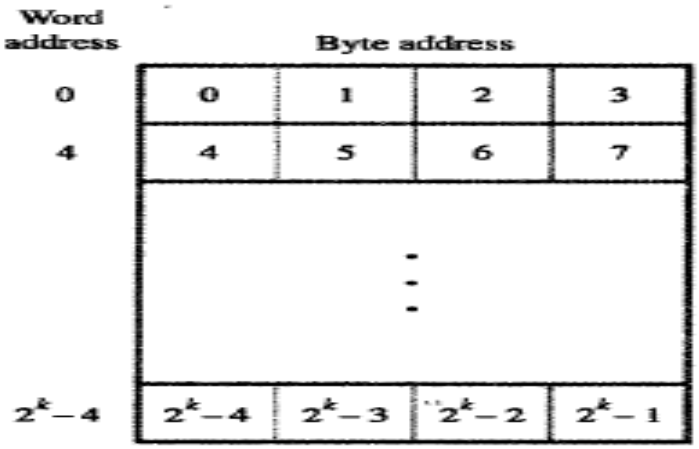
B_{31}	B_{30}	……	B_1	B_0

8 bits	8 bits	8 bits	8 bits
ASCII characters	**ASCII characters**	**ASCII characters**	**ASCII characters**

BIG ENDIAN ASSIGNMENT

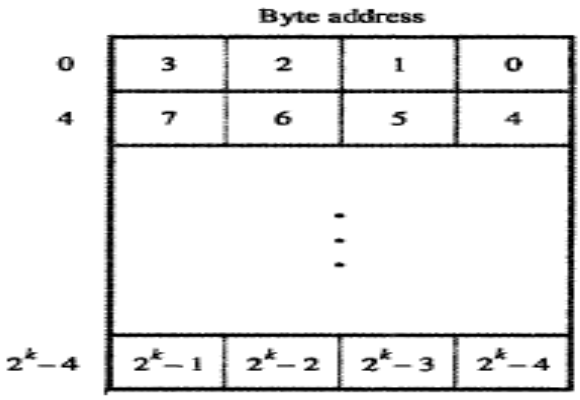
In this technique lower byte of data is assigned to higher address of the memory and higher byte of data is assigned to lower address of the memory.

The structure of memory to represent 32 bit number for Big Endian assignment is as shown in the fig.

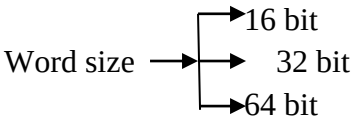


LITTLE ENDIAN ASSIGNMENT

In this technique lower byte of data is assigned to lower address of the memory and higher byte of data is assigned to higher address of the memory.
The structure of memory to represent 32 bit number for little endian assignment is as shown in the fig.



WORD ALLIGNMENT



It is process of assignment of addresses of two successive words and this address is the number of bytes in the word is called as Word alignment.

ACCESSING CHARACTERS AND NUMBERS

The character occupies 1 byte of memory and hence byte address for memory.
The numbers occupies 2 bytes of memory and hence word address for numbers.

➢ *MEMORY OPERATION*

There are two types of memory operations namely 1. Memory read and 2. Memory write

MEMORY READ OPERATION:

- It is the process of transfering of 1 word of data from memory into Accumulator.
- It is also called as Memory fetch operation.

- The Memory read operation can be implemented by means of LOAD instruction. The LOAD instruction transfers 1 word of data (1 word = 32 bits) from Memory into the Accumulator .

MEMORY WRITE OPERATION

- It is the process of transferring the 1 word of data from Accumulator into the memory.
- The Memory write operation can be implemented by means of STORE instruction. The STORE instruction transfers 1 word of data from Accumulator into the Memory location as shown in the fig.

➢ *INSTRUCTION AND INSTRUCTION SEQUENCING*

The Computer is designed to perform 4 types of operations, namely

- Data transfer operations
- ALU Operations
- Program sequencing and control.
- I/O Operations.

REGISTER TRANSFER NOTATION

- There are 3 locations to store the operands during the execution of the program namely 1. Register 2. Memory location 3. I/O Port. Location is the storage space used to store the data.
- The instructions are designed to transfer data from one location to another location. Consider the first statement to transfer data from one location to another location
- " Transfer the contents of Memory location whose symbolic name is given by AMOUNT into processor register R_0."
- The mathematical representation of this statement is given by

  $R_0 \leftarrow [AMOUNT]$

  Consider the second statement to add data between two registers
- "Add the contents of R_0 with the contents of R_1 and result is stored in R_2"
- The mathematical representation of this statement is given by

  $R_2 \leftarrow [R_0] + [R_1]$.

Such a notation is called as "Register Transfer Notation".

It uses two symbols

1. A pair of square brackets [] to indicate the contents of Memory location and
2. ← to indicate the data transfer operation.

ASSEMBLY LANGUAGE NOTATION

Consider the first statement to transfer data from one location to another location

- " Transfer the contents of Memory location whose symbolic name is given by AMOUNT into processor register R_0."
- The assembly language notation of this statement is given by

MOVE AMOUNT, R_0
Opcode Source Destination

This instruction transfers 1 word of data from Memory location whose symbolic name is given by AMOUNT into the processor register R_0.

- The mathematical representation of this statement is given by

$R_0 \leftarrow [AMOUNT]$

Consider the second statement to add data between two registers

- "Add the contents of R_0 with the contents of R_1 and result is stored in R_2"
- The assembly language notation of this statement is given by

ADD R_0 , R_1, R_2
Opcode source1, Source2, Destination

This instruction adds the contents of R_0 with the contents of R_1 and result is stored in R_2.

- The mathematical representation of this statement is given by

$R_2 \leftarrow [R_0] + [R_1]$.

Such a notations are called as "Assembly Language Notations"

BASIC INSTRUCTION TYPES

There are 3 basic instructions namely

1. Three address instruction format
2. Two address instruction format
3. One address instruction format

Consider the arithmetic expression C = A + B, Where A,B,Z are the Memory locations.

Steps for evaluation

1. Access the first memory operand whose symbolic name is given by A.
2. Access the second memory operand whose symbolic name is given by B.
3. Perform the addition operation between two memory operands.
4. Store the result into the 3rd memory location C.
5. The mathematical representation is $C \leftarrow [A] + [B]$.

a) Three address instruction format : Its format is as follows

opcode	Source-1	Source-2	destination

Destination ← [source-1] + [source-2]
Ex: ADD A, B,C
C ← [A] + [B]

a) Two address instruction format : Its format is as follows

opcode	**Source**	**destination**

Destination ← [source] + [destination]
The sequence of two address m/c instructions to evaluate the arithmetic expression
C ← A + B are as follows

MOV A, R_0
MOV B, R_1
ADD R_0, R_1
MOV R_1, C

b) One address instruction format : Its format is as follows

opcode	**operand**

Ex1: LOAD A This instruction copies the contents of memory location whose symbolic name is given by 'A' into the Accumulator as shown in the figure.
ADD B This instruction adds the contents of Accumulator with the contents of Memory location 'B' and result is stored in Accumulator.
STORE B This instruction copies the contents of Accumulator into memory location whose symbolic name is given by 'B' .

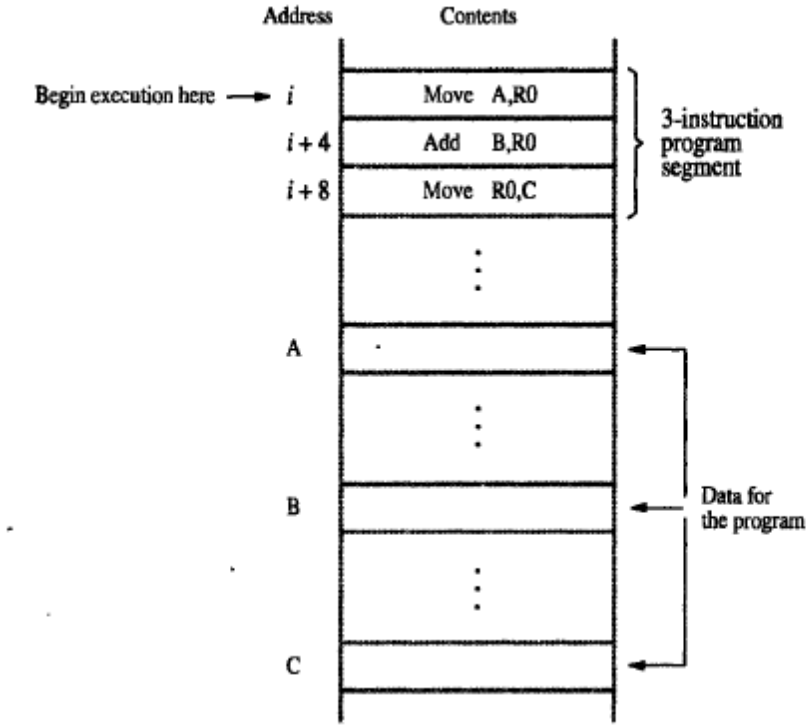
INSTRUCTIONS EXECUTION AND STRAIGHT LINE SEQUENCING

Consider the arithmetic expression
C = A+B , Where A,B,C are the memory operands.
The mathematical representation of this instruction is
C = [A] + [B].
The sequence of instructions using two address instruction format are as follows

MOV A, R_0
ADD B, R_0
MOV R_0, C

Such a program is called as 3 instruction program.
NOTE: The size of each instruction is 32 bits.

The 3 instruction program is stored in the successive memory locations of the processor is as shown in the fig.



The system bus consists of uni-directional address bus,bi-directional data bus and control bus. "It is the process of accessing the 1st instruction from memory whose address is stored in program counter into Instruction Register (IR) by means of bi-directional data bus and at the same time after instruction access the contents of PC are incremented by 4 in order to access the next instruction. Such a process is called as "Straight Line Sequencing".

INSTRUCTION EXECUTION

There are 4 steps for instruction execution

1 Fetch the instruction from memory into the Instruction Register (IR) whose address is stored in PC.

IR ← [[PC]]

2 Increment the contents of PC by 4.

PC ← [PC] + 4.

3 Decode the instruction.

4 Perform the operation according to the opcode of an instruction

5 Load the result into the destination.

BRANCHING

- Instead of using a long list of add instructions, it is possible to place a single Add instruction in a program loop as shown in figure.
- The loop is a straight- line sequence of instructions executed as many times as needed.

- It starts at location LOOP and ends at the instruction Branch>0.
- During each pass through this loop, the address of the next list entry is determined, and that entry is fetched and added to R0.
- Assume that the number of entries in the list , n is stored in memory location N, as shown. Register R1 is used as a counter to determine the number of times the loop is executed.
- Hence the contents of location n are loaded into register R1 at the beginning of the program. Then, within the body of the loop, the instruction
    - Decrement R1
- Reduces the contents of R1 by each time through the loop.
- Execution of the loop is repeated as long as the result of the decrement operation is greater than Zero.

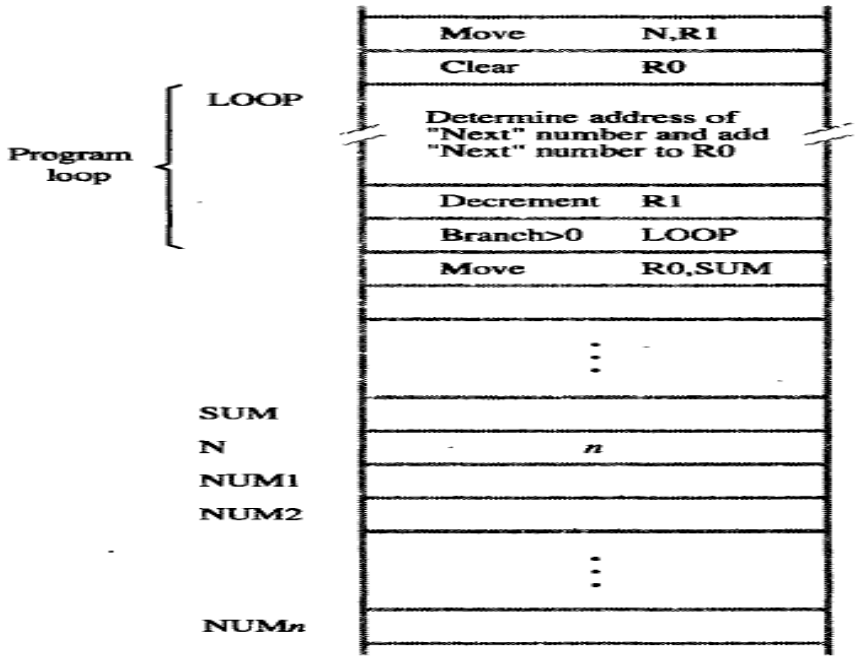


Figure 2.10 Using a loop to add *n* numbers.

CONDITION CODE BITS

- The processor consists of series of flip-flops to store the status information after ALU operation.
- The series of flip-flip-flops used to store the status and control information of the processor is called as "Condition Code Register". It defines 4 flags. The format of condition code register is as follows.

C	V	Z	N

1 N (NEGATIVE) Flag:
It is designed to differentiate between positive and negative result.
It is set 1 if the result is negative, and set to 0 if result is positive.

2 Z (ZERO) Flag:
It is set to 1 when the result of an ALU operation is found to zero, otherwise it is cleared.

3 V (OVER FLOW) Flag:
In case of 2^s Complement number system n-bit number is capable of representing a range of numbers and is given by -2^{n-1} to $+2^{n-1.}$. The Over-Flow flag is set to 1 if the result is found to be out of this range.

4 C (CARRY) Flag :
This flag is set to 1 if there is a carry from addition or borrow from subtraction, otherwise it is cleared..

➢ *ADDRESSING MODES*

The different ways in which the location of an operand is pecified in an instruction are referred to as addressing modes

IMPLEMENTATION OF VARIABLES AND CONSTANTS

1.REGISTER ADDRESSING

In this mode operands are stored in the registers of CPU. The name of the register is directly specified in the instruction.

Ex: MOVE R1,R2 Where R1 and R2 are the Source and Destination registers respectively. This instruction transfers 32 bits of data from R1 register into R2 register. This instruction does not refer memory for operands. The operands are directly available in the registers.

2. ABSOLUTE ADDRESSING

It is also called as Absolute Addressing Mode. In this addressing mode operands are stored in the memory locations. The name of the memory location is directly specified in the instruction.

Ex: MOVE X, R_1 : Where X is the memory location and R1 is the Register.

This instruction transfers 32 bits of data from memory location X into the General Purpose Register R1.

3.IMMEDIATE ADDRESSING

In this Addressing Mode operands are directly specified in the instruction. The source field is used to represent the operands. The operands are represented by # (hash) sign.

Ex: MOVE #23, R0

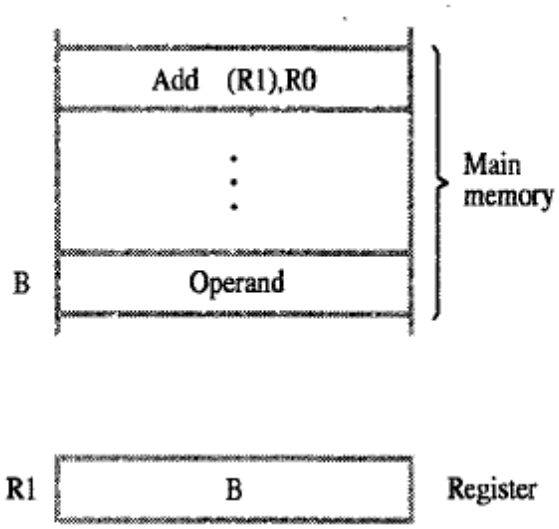
INDIRECTION AND POINTERS

i)Indirect Addressing Mode through GPR·

In this Addressing Mode the effective address of an operand is stored in the one of the General Purpose Register of the CPU.

Ex: ADD (R1), R0 ; Where R1 and R0 are GPRs.

This instruction adds the data from the memory location whose address is stored in R1 with the contents of R0 Register and the result is stored in R0 register as shown in the fig.



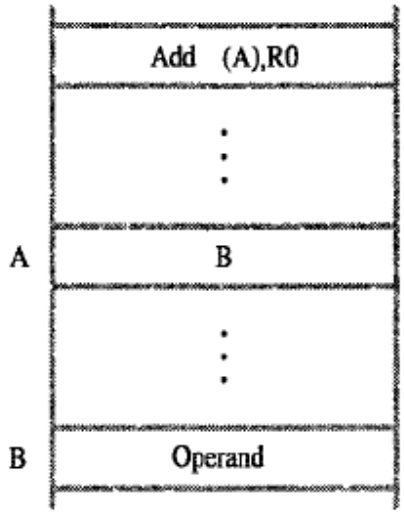
ii) Indirect Addressing Mode through Memory Location

In this Addressing Mode, effective address of an operand is stored in the memory location.

Ex: ADD (A), R_0

This instruction adds the data from the memory location whose address is stored in 'A' memory location with the contents of R_0 and result is stored in R_0 register.

The diagrammatic representation of this addressing mode is as shown in the fig.



INDEXING AND ARRAYS

In this addressing mode, the effective address of an operand is computed by adding constant value with the contents of Index Register and any one of the General Purpose Register namely R0 to Rn-1 can be used as the Index Register. The constant value is directly specified in the instruction.

The symbolic representations of this mode are as follows

X (Ri) where X is the Constant value and Rj is the GPR.

It can be represented as

EA of an operand = X + (R_i)

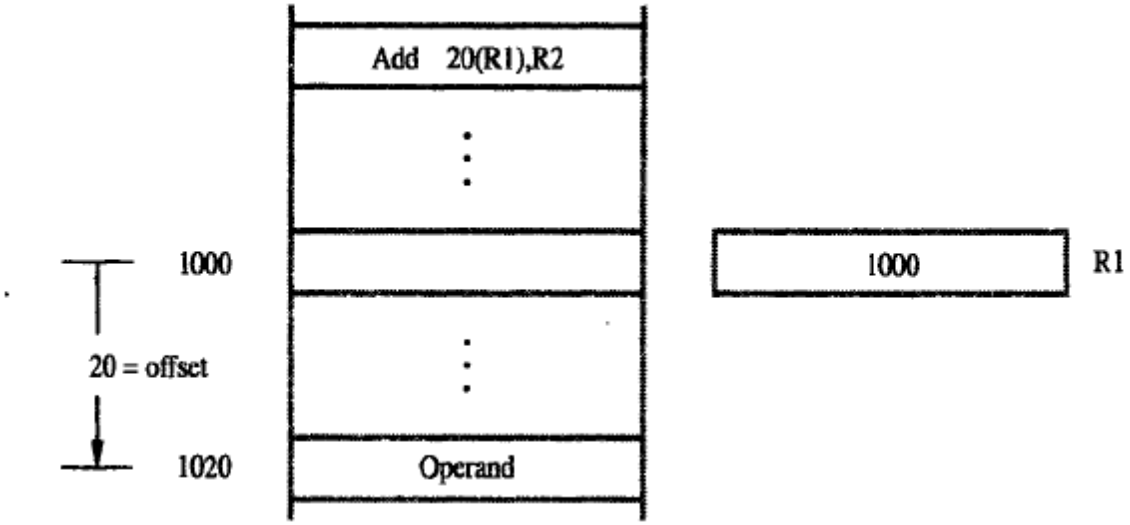
i)Offset is given as constant

Ex: ADD 20(R_1), R_2

The EA of an operand is given by

EA = 20 + [R_1]

This instruction adds the contents of memory location whose EA is the sum of contents of R_1 with 20 and with the contents of R_2 and result is placed in R_2 register. The diagrammatic representation of this mode is as shown in the fig.



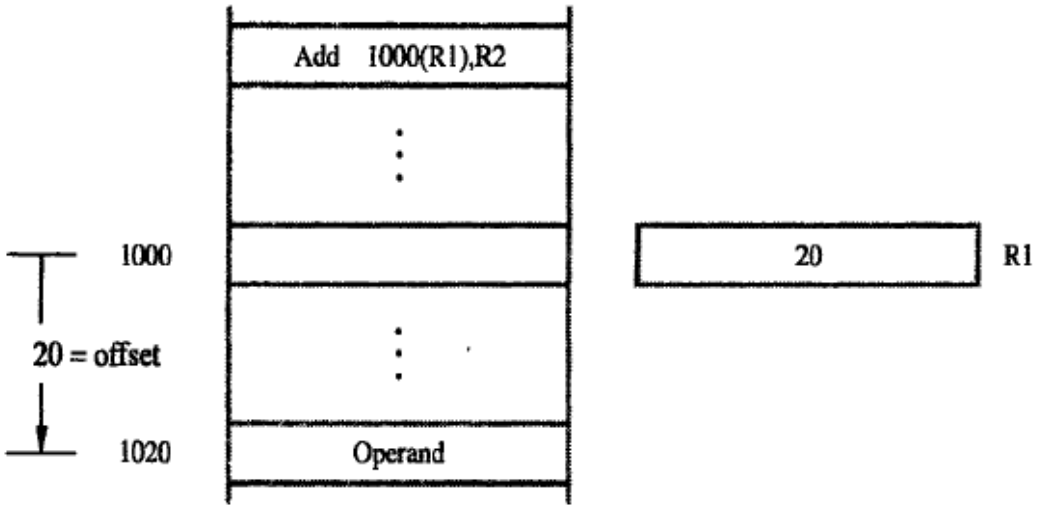
ii) Offset is in Index Register

Ex: ADD 1000(R_1) , R_2 R_1 holds the offset address of an operand.

The EA of an operand is given by

EA = 1000 + [R_1]

This instruction adds the data from the memory location whose address is given by 1000 + [R1] with the contents of R_2 and result is placed in R_2 register.



RELATIVE ADDRESSING

In this Addressing Mode EA of an operand is computed by the Index Addressing Mode. This Addressing Mode uses PC (Program Counter)to store the EA of the next instruction instead of GPR.The symbolic representation of this mode is X (PC).Where X is the offset value and PC is the Program Counter to store the address of the next instruction to be executed.

It can be represented as

EA of an operand = X + (PC).

This Addressing Mode is useful to calculate the EA of the target memory location.

ADDITIONAL MODES

AUTOINCREMENT ADDRESSING MODE

In this Addressing Mode , EA of an operand is stored in the one of the GPR^s of the CPU. This Addressing Mode increment the contents of memory register by 4 memory locations after operand access.

The symbolic representation is

$(R_I)+$ Where R_i is the one of the GPR.

Ex: MOVE $(R_1)+$, R_2

This instruction transfer's data from the memory location whose address is stored in R_1 into R_3 register and then it increments the contents of R_1 by 4 memory locations.

AUTODECREMENT ADDRESSING MODE

In this Addressing Mode , EA of an operand is stored in the one of the GPR^s of the CPU. This Addressing Mode decrements the contents of memory register by 4 memory locations and then transfers the data to destination.

The symbolic representation is

$-(R_I)$ Where R_i is the one of the GPR.

Ex: MOVE $-(R_1)$, R_2

This instruction first decrements the contents of R_1 by 4 memory locations and then transfer's data to destination register.